Grado en Ingeniería Informática
Grado en Matemáticas e Informática

Asignatura: PROGRAMACIÓN II

Variables, Valores y Referencias

Profesores de Prog II

DLSIIS - E.T.S. de Ingenieros Informáticos
Universidad Politécnica de Madrid

Febrero 2015

Tipos básicos en Java

- Java maneja algunos datos que no son objetos
- Tipos básicos:
  - **`int`**: valores numéricos enteros
  - **`float, double`**: valores numéricos con parte fraccionaria
  - **`boolean`**: valores lógicos (**`true, false`**)
  - **`char`**: valores de tipo carácter

Variables en Java

- Una variable es un contenedor que puede almacenar información
- Una variable tiene siempre asociado un tipo o clase de valores
- Una variable se declara indicando su tipo, nombre y (opcionalmente) el valor inicial

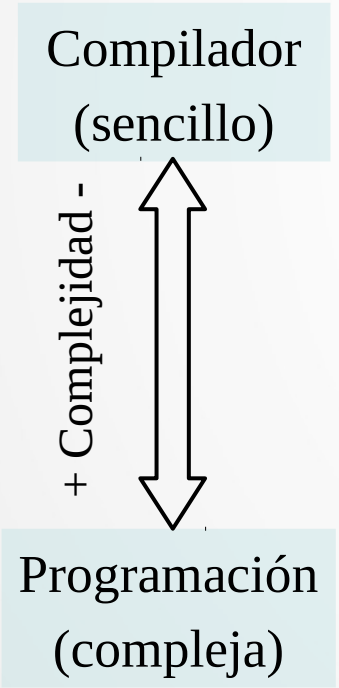
```
int anio;
double longitud = 0.0;
float precio = 24.12f ;
Fecha inauguracion;
Fecha inicioCurso = new Fecha( 3, 9, 2013);
```

Puntero o Referencia

- Las máquinas tienen la memoria dividida en celdas, cada celda tiene **una dirección y un contenido**.
- El contenido dependerá del programa:
  - Datos de tipo entero
  - Datos de tipo booleano
  - Datos de tipo array
  - Código de programa
- **Clave**: incluso el contenido de una celda puede ser una dirección de memoria → **Puntero**
- En java todas las variables de tipo clase (instancias) son en realidad punteros (referencias a objetos)

Tipos de Memoria

- Clasificaciones diferentes de la memoria:
  - Memoria **estática** vs **dinámica**
  - Memoria **automática** vs **no automática**



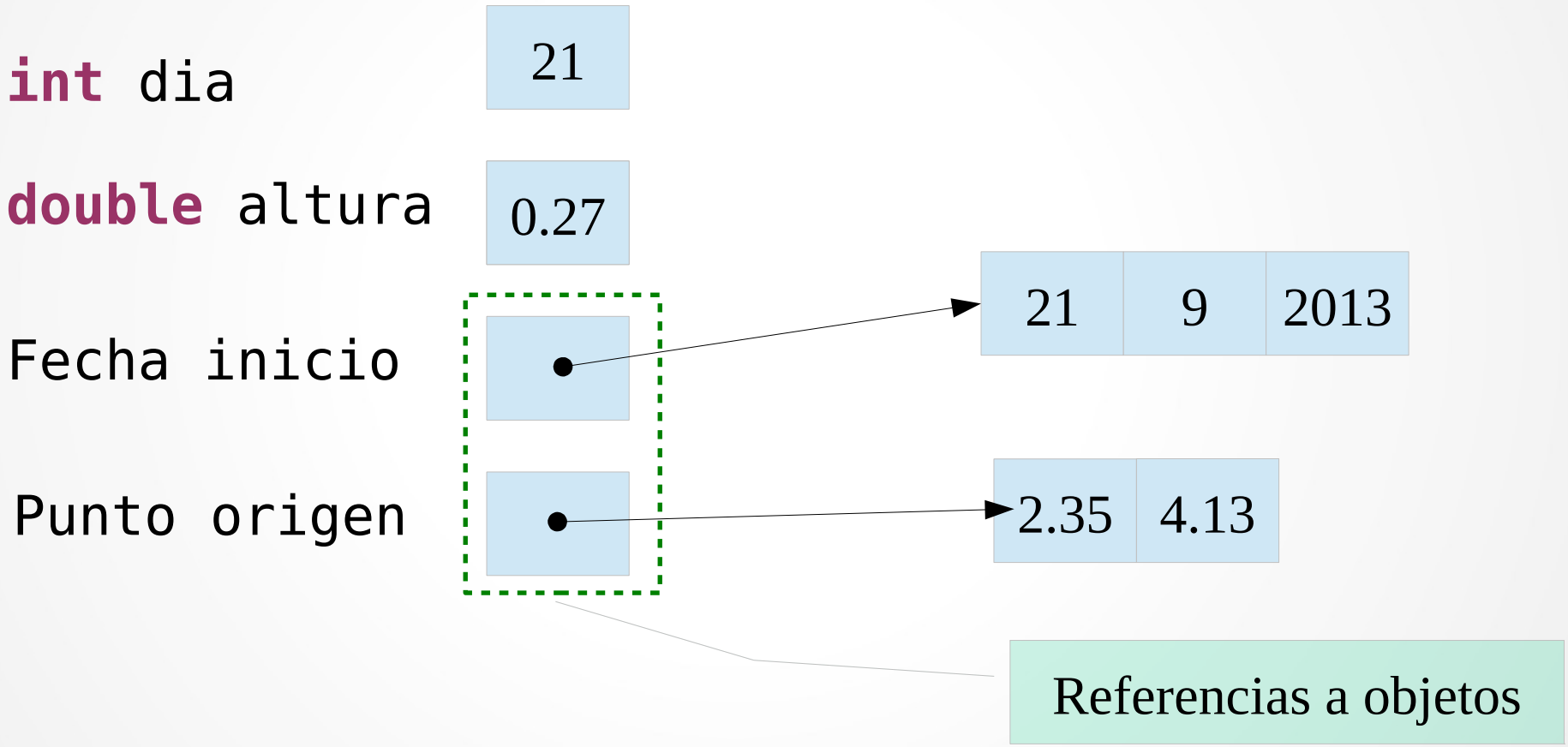
Tipo	Automático	No automático
Estática	Variables Globales	-------
Dinámica	Parámetros Formales Variables Locales	Referencias (o punteros)

Tipos de Memoria

- Clasificaciones diferentes de la memoria:
  - Tendencias:
    - Todo automático: Programación funcional, lógica,
    - No automática:
      - **Administrada**: Solicitud de memoria por el programador y recolector de memoria no usada (Java, .Net) → **GC**
      - **No administrada**: C, C++, Ada, ...

Valores y referencias (punteros)

- Una variable de un **tipo básico** contiene un **valor**
- Una **variable objeto** contiene una **referencia al objeto**



Uso de valores y referencias

- Las variables que contienen valores se comportan de forma diferente a las variables que contienen referencias, tales como:
  - Asignación:

    `=`
  - Comparación por igualdad:

    `==`
  - Paso de argumentos a métodos:

```
operacion(arg1, arg2)
```

Asignación, ¿copia valor o referencia?

```
public static void main(String[] args) {
  int numero1, numero2;
  Punto punto1, punto2;

  numero1 = 3;
  numero2 = numero1;  // copia valor
  System.out.println("numero1=" + numero1 + "; numero2=" + numero2);
  numero1 = 5;
  System.out.println("numero1=" + numero1 + "; numero2=" + numero2);

  punto1 = new Punto(3, 4);
  punto2 = punto1;    // copia referencia
  System.out.println("punto1=" + punto1 + "; punto2=" + punto2);
  punto1.setX(9);
  System.out.println("punto1=" + punto1 + "; punto2=" + punto2);
}
```

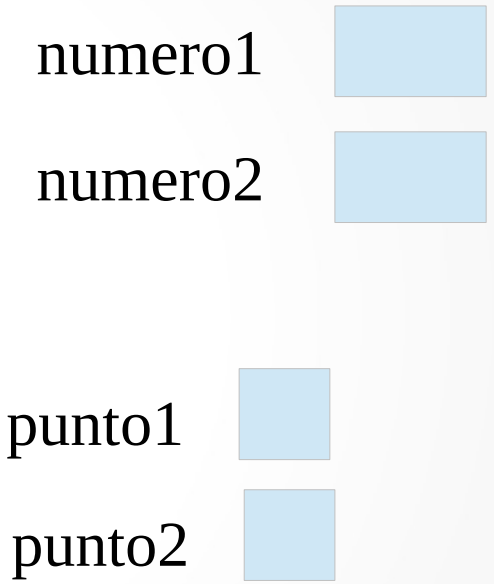
```
numero1=3; numero2=3
numero1=5; numero2=3
punto1=(3.0,4.0); punto2=(3.0,4.0)
punto1=(9.0,4.0); punto2=(9.0,4.0)
```

Asignación, ¿copia valor o referencia?

```
int numero1, numero2;
Punto punto1, punto2;

numero1 = 3;
numero2 = numero1;  // copia valor

numero1 = 5;


punto1 = new Punto(3, 4);
punto2 = punto1;    // copia referencia

punto1.setX(9);
```

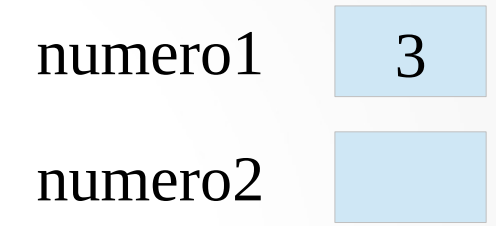
Asignación, ¿copia valor o referencia?

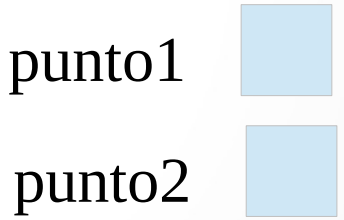
```
int numero1, numero2;
Punto punto1, punto2;

numero1 = 3;
numero2 = numero1;  // copia valor

numero1 = 5;


punto1 = new Punto(3, 4);
punto2 = punto1;    // copia referencia

punto1.setX(9);
```



Asignación, ¿copia valor o referencia?

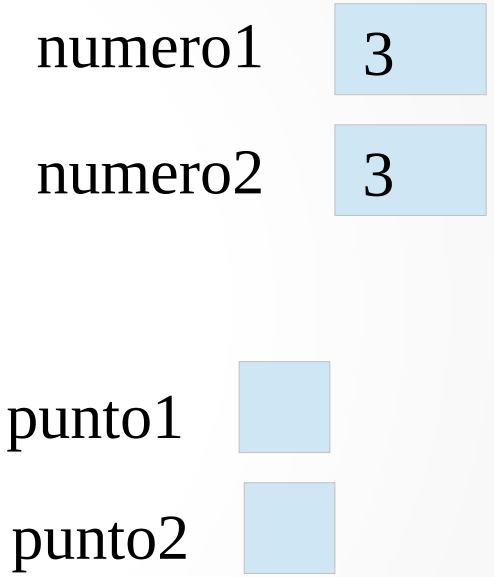
```
int numero1, numero2;
Punto punto1, punto2;

numero1 = 3;
numero2 = numero1;  // copia valor

numero1 = 5;


punto1 = new Punto(3, 4);
punto2 = punto1;    // copia referencia

punto1.setX(9);
```





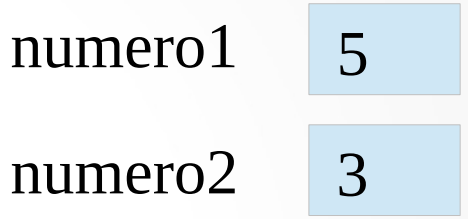
Asignación, ¿copia valor o referencia?

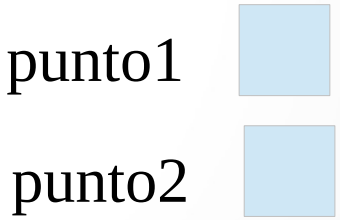
```
int numero1, numero2;
Punto punto1, punto2;

numero1 = 3;
numero2 = numero1;  // copia valor

numero1 = 5;


punto1 = new Punto(3, 4);
punto2 = punto1;    // copia referencia

punto1.setX(9);
```



Asignación, ¿copia valor o referencia?

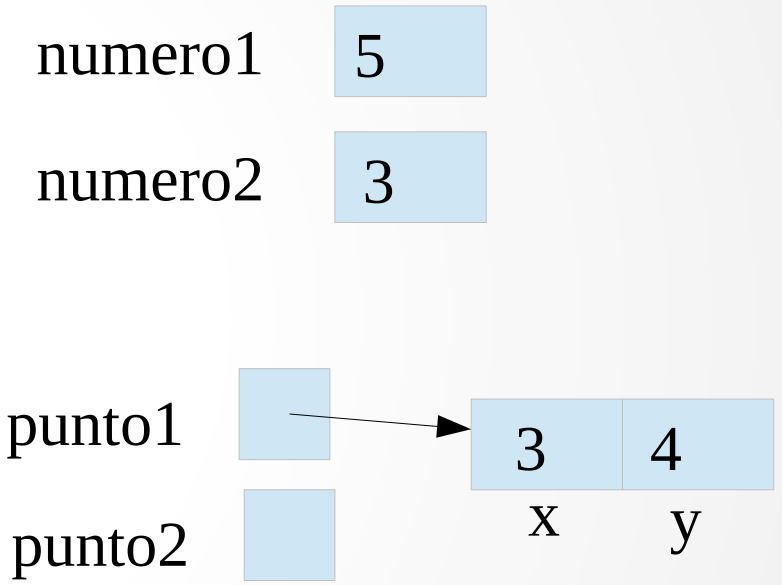
```
int numero1, numero2;
Punto punto1, punto2;

numero1 = 3;
numero2 = numero1;  // copia valor

numero1 = 5;


punto1 = new Punto(3, 4);
punto2 = punto1;    // copia referencia

punto1.setX(9);
```





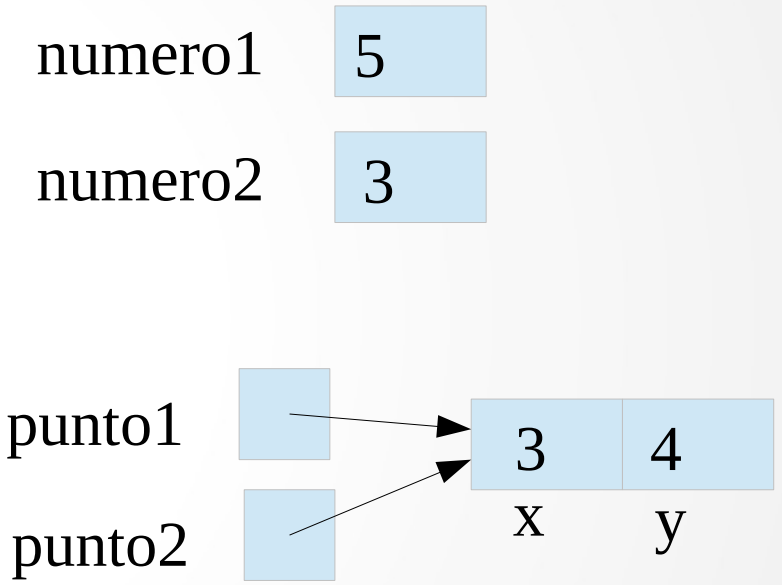
Asignación, ¿copia valor o referencia?

```
int numero1, numero2;
Punto punto1, punto2;

numero1 = 3;
numero2 = numero1;  // copia valor

numero1 = 5;


punto1 = new Punto(3, 4);
punto2 = punto1;    // copia referencia

punto1.setX(9);
```



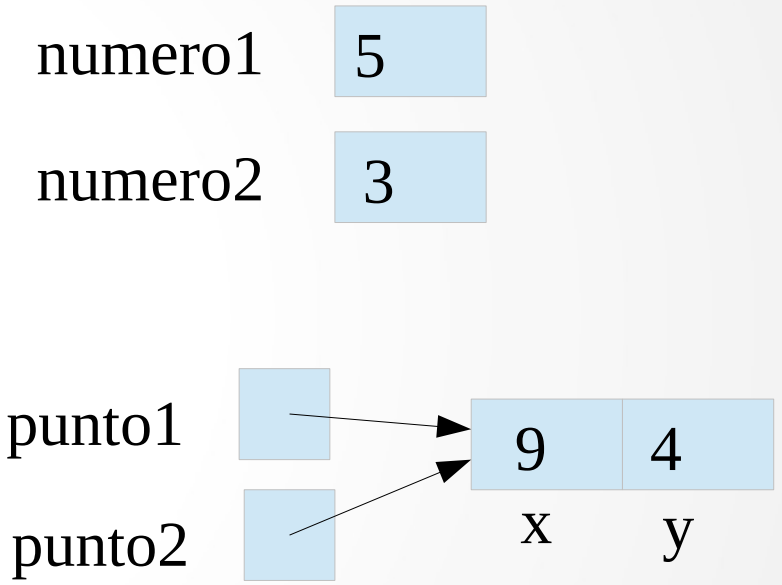
Asignación, ¿copia valor o referencia?

```
int numero1, numero2;
Punto punto1, punto2;

numero1 = 3;
numero2 = numero1;  // copia valor

numero1 = 5;


punto1 = new Punto(3, 4);
punto2 = punto1;    // copia referencia

punto1.setX(9);
```



Asignación, ¿copia valor o referencia?

```
int numero1, numero2;
Punto punto1, punto2;

numero1 = 3;
numero2 = numero1;  // copia valor

numero1 = 5;


punto1 = new Punto(3, 4);
punto2 = punto1;    // copia referencia

punto1.setX(9);
```



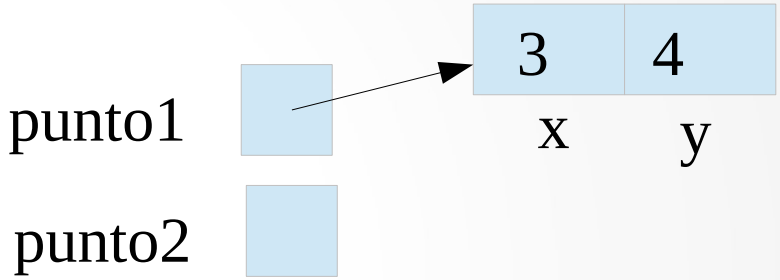
Asignar valor de un objeto

- Para copiar objetos (su valor) hay que crear un duplicado
- Para ello se puede usar un constructor de copia

```
public Punto(Punto p) {    // constructor de copia
   x = p.x;
   y = p.y;
}

....
   punto1 = new Punto(3, 4);
   punto2 = new Punto(punto1);    // copia valor (duplicado)
   System.out.println("punto1=" + punto1 + "; punto2=" + punto2);
   punto1.setX(9);
   System.out.println("punto1=" + punto1 + "; punto2=" + punto2);
....
```

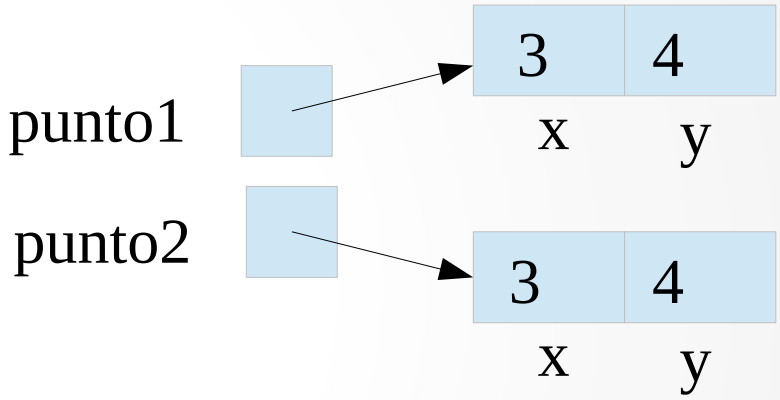
Asignar valor de un objeto

```
punto1 = new Punto(3, 4);
punto2 = new Punto(punto1);

punto1.setX(9);
```



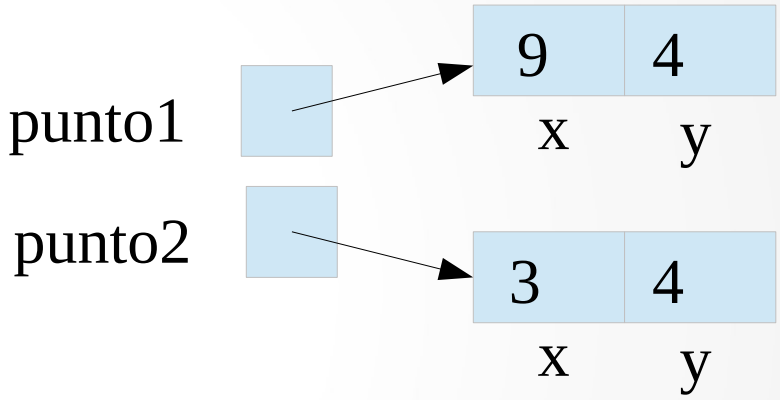
Asignar valor de un objeto

```
punto1 = new Punto(3, 4);
punto2 = new Punto(punto1);

punto1.setX(9);
```



Asignar valor de un objeto

```
punto1 = new Punto(3, 4);
punto2 = new Punto(punto1);

punto1.setX(9);
```



Igualdad

- El operador **==** compara el contenido directo de las variables
- Por tanto, si las variables son objetos, compara referencias

```
Punto punto1, punto2;

punto1 = new Punto(3, 4);
punto2 = punto1;     // copia referencia
System.out.println("punto1==punto2: " + (punto1==punto2));
    // cierto

punto2 = new Punto(punto1); // copia valor (duplicado)
System.out.println("punto1==punto2: " + (punto1==punto2));
    //¡falso!
```

- En Java el operador **==** representa la identidad de objetos

Igualdad

- Para evaluar si dos objetos contienen los mismos valores hay que definir un método de comparación

```
public class Punto {
  private double x, y;
....
  // Puntos iguales si tienen las mismas coordenadas
  public boolean esIgual (Punto p) {
    return x == p.x && y == p.y;
  }
....
}
```

Igualdad

- La función `esIgual()` representa la equivalencia entre puntos

```
 Punto punto1, punto2;
punto1 = new Punto(3, 4);
punto2 = punto1;    // copia referencia
System.out.println(
  "punto1.esIgual(punto2): " + punto1.esIgual(punto2)); // cierto

punto2 = new Punto(punto1); // copia valor (duplicado)
System.out.println(
  "punto1.esIgual(punto2): " + punto1.esIgual(punto2)); // cierto
```

Ejercicio II

- Añade al fichero TestPunto pruebas en las que compares puntos utilizando el método esI*gual*

Comparar objetos que contienen objetos

- Si un atributo es un objeto, hay que compararlo también con la correspondiente función esIgual() o similar

```
public class Flecha {
  private Punto inicio, final;
....
  // Flechas iguales si tienen los mismos extremos
  public boolean esIgual(Flecha fl) {
    return inicio.esIgual(fl.inicio) &&
           final.esIgual(fl.final);
  } // de esIgual
....
} // de Flecha
```

Paso de parámetros

- Algunos lenguajes de programación (C++, ADA, MODULA-2, etc.) pueden utilizar dos formas alternativas de pasar los parámetros al llamar a un subprograma:
  - **Paso por valor** (para los datos de entrada que recibe el subprograma)
    - Los datos de entrada no deben ser modificados
    - Se saca una copia del dato original que se pasa como argumento
    - Se tienen 2 valores duplicados e independientes, por tanto los cambios en uno no afectan al otro
  - **Paso por referencia** (para los datos de entrada/salida o salida)
    - Los datos van a ser inicializados o modificados dentro del subprograma
    - Se pasa una referencia al dato original
    - Cualquier acción implica cambio en los 2 valores (el que está fuera y el que está dentro del subprograma)

Paso de parámetros en Java

- El paso de parámetros en Java sólo es por valor
  - Se hace una copia del argumento en el contexto del método.
- **Pero, OJO:**
  - En Java las **variables de tipo objeto** realmente **contienen referencias a objetos**
  - Si pasamos como argumento un objeto, pasamos una copia de una referencia a él.
  - Los cambios en los objetos afectan **a todas sus referencias**
  - El objeto que se pasa como argumento se puede modificar dentro del método (dato de entrada/salida o salida)

Ejemplo: intercambiar dos variables

- No se puede implementar un método que intercambie dos argumentos de los tipos básicos

```
static void intercambiar(int p, int q) {
  int aux = p;
  p = q;
  q = aux;
}

public static void main(String[] args) {
  int a=3; int b=7;
  System.out.println(a + " " + b);  // --> 3 7

  int aux = a;
  a = b;
  b = aux;
  System.out.println(a + " " + b);  // --> 7 3

  intercambiar(a,b);  // no intercambia
  System.out.println(a + " " + b);  // --> 7 3
}
```

Ejemplo: intercambiar dos variables

- El esquema de código anterior tampoco funciona con argumentos que sean objetos (referencias)

```
static void intercambiar(Punto p, Punto q) {
  Punto aux = p;
  p = q;
  q = aux;
}

public static void main(String[] args) {
  Punto a = new Punto(3,7);
  Punto b = new Punto(4,8);
  System.out.println(a + " " + b);  // --> (3,7) (4,8)
  Punto aux = a;
  a = b;
  b = aux;
  System.out.println(a + " " + b);  // --> (4,8) (3,7)
  intercambiar(a,b);  // no intercambia
  System.out.println(a + " " + b);  // --> (4,8) (3,7)
}
```

Ejemplo: intercambiar dos variables

- Para intercambiar objetos hay que intercambiar sus contenidos

```
static void intercambiar(Punto p, Punto q) {
  double xx = p.x; p.x = q.x; q.x = xx;
  double yy = p.y; p.y = q.y; q.y = yy;
}

public static void main(String[] args) {
  Punto a = new Punto(3,7);
  Punto b = new Punto(4,8);
  System.out.println(a + " " + b);  // --> (3,7) (4,8)
  Punto aux = a;
  a = b;
  b = aux;
  System.out.println(a + " " + b);  // --> (4,8) (3,7)
  intercambiar(a,b);  // ahora sí intercambia
  System.out.println(a + " " + b);  // --> (3,7) (4,8)
}
```